Sustainable Quality of Service for real-time jobs in Autonomous Computing Devices

Rola El Osta, Maryline Chetto

Sustainable Quality of Service for real-time jobs in Autonomous Computing Devices

Rola El Osta, and Maryline Chetto
University of Nantes
IRCCyN, UMR CNRS 6597, 44321 Nantes, France
Email: firstname.lastname@irccyn.ec-nantes.fr

Abstract

Energy harvesting has emerged as an efficient way for powering wireless devices in order to extend their lifetime. In this paper a dynamic power management strategy is described that guarantees the correct execution of every feasible application for a given energy harvester, energy storage unit and processor. The so-called ED-H strategy was recently proved to be optimal in that sense that ED-H optimizes the Quality of Service defined as the ratio of deadline success for the real-time jobs. We propose to describe this novel strategy for dynamic power management and scheduling.

Index Terms

Dynamic power management, uniprocessor device, real-time scheduling, energy harvesting

1. Introduction

The lifetime of individual sensors in wireless sensor networks (WSN) highly depends on batteries which are limited because of size, weight and other physical requirements. In order to improve sustainability of WSNs, an approach known as energy harvesting consists in using rechargeable batteries or capacitors in order to store the energy which is harvested from the environment. Energy harvesting now permits sensors to run for very long periods of time (even perpetually) with no need for periodic battery replacement. This is a particular attractive technology when battery replacement is impractical due to deployment conditions. The energy can be drawn from various ambient sources such as light, radio frequency, thermal gradients, motion movements [12], etc.

However, the design of a sensor node that uses energy harvesting raises specific problems which are different from those with the classical power sources [7]. The first problem lies in that the environmental energy sources do not provide constant power i.e. the harvested energy can vary significantly over time which leads to either energy starvations or energy overflows. The second problem is that the energy storage unit has a finite capacity.
In this paper, we address the scheduling problem for a single processor device that executes preemptible time critical jobs. Each one has a certain energy requirement and has to execute by a certain deadline. A job can be the invocation of a periodic task or sporadic task or it can be aperiodic , thus arriving at an unpredictable time. The scheduling problem we have to deal with is to guarantee all the timing requirements of the jobs by suitably exploiting both the processor capacity and the available ambient energy.

The aim of this paper is to describe a new strategy which permits to decide when to execute a job and which job to execute. The Earliest Deadline for energy Harvesting systems scheduling algorithm, ED-H for short, recently proved to be optimal [3], is consequently for both dynamic power management of the computing device and real-time scheduling of jobs.

Outline The remainder of the paper is organized as follows. Related work and background materials on scheduling and overload management are presented in Section II and section III. In section IV, we present the system model. Section V describes concepts and the novel ED-H scheduler. Main results about ED-H are summarized in section VI. Section VII focuses on practical considerations.Section VIII gives concluding remarks.

978-1-4799-4357-9/14 $31.00 © 2014 IEEE
DOI 10.1109/FiCloud.2014.81

2. Related work

In a Real-Time Energy Harvesting (RTEH) system, it is sometimes preferable even necessary to let the processor inactive and not to execute a ready job. This is because energy starvation could prevent future occurring jobs to meet their deadlines. Another key consideration is that the system must operate with energy neutrality, thus consuming only as much energy as harvested [6].

The Lazy Scheduling algorithm, known as LSA, provides an optimal solution to the scheduling problem in RTEH systems [11]. However, this algorithm assumes that the energy demand of a job and its actual execution time behave proportionally. Such a scheduling solution cannot be applied in most of systems where instantaneous power consumed by jobs varies along time depending on circuitry and devices required by their execution. This observation has consequently motivated our work on a more realistic energy model considered hereafter.

Initially, we investigated the issue of scheduling periodic task sets only in [4]. We presented an heuristic called EDeg (Earliest Deadline with energy guarantee). However we did not provide schedulability test and formal performance evaluation for it. EDeg has been compared to EDF (Earliest Deadline First) scheduler through simulations. It makes a significant performance enhancement in comparison to a classical non idling and non clairvoyant scheduler such as EDF. In [5], we also proposed an online scheduler called EH-EDF which is another variant of EDF. EH-EDF is non-clairvoyant but an idling scheduler that permits the processor to stay inactive despite some pending jobs. EH-EDF outperforms EDF due to its idling capabilities. Research on fixed priority scheduling for RTEH systems has been reported in [1]. An optimal scheduler called PFPasap was proposed for periodic task sets and a constant source power only.

3. Background Materials

Real-time scheduling theory mainly deals with feasibility analysis and online priority driven scheduling [8]. EDF is an online non-idling dynamic-priority scheduling algorithm which at each instant chooses for execution the job with the closest absolute deadline. EDF is optimal for scheduling arbitrary collections of independent jobs [9] with no energy limitations. The jobs may result from invocations of periodic tasks, invocations of sporadic tasks or they can be aperiodic jobs. By virtue of its optimality, any feasible set of jobs

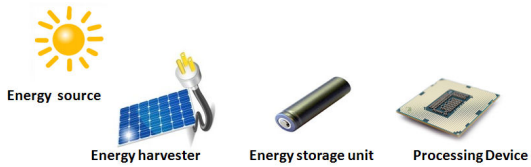

Figure 1. An Energy Harvesting System

is guaranteed to be successfully scheduled by EDF i.e. all the deadlines will be met in the EDF schedule.

4. The system model

In that paper, our work focusses on a RTEH system which comprises one processing module (PM), an energy harvesting module (HM) and an energy storage module (SM) (see Fig. 1). The energy consumption of the PM is only due to dynamic switching energy and the PM is supplied exclusively with energy generated by the environmental source. The real-time jobs need to be executed on the PM before their respective deadline. Each job is characterized by four parameters - arrival time, worst case execution time, worst case energy requirement and deadline.

At every time t, the HM harvests energy from the environment and converts it into electrical power with instantaneous charging rate $P_p(t)$. The energy harvested in the time interval $[t_1, t_2)$ is $E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$. We can predict the incoming energy accurately for near future. We assume that the electrical energy produced by the HM in any unit of time never exceeds the energy consumed in that unit of time.

Our system uses an energy storage energy unit (e.g. super-capacitor or rechargeable battery) to continue operation even when there is no energy to harvest. Its nominal capacity C corresponds to the maximum amount of energy (expressed in energy unit) that can be stored at any time. The SM receives power from the HM and delivers power to the PM. The stored energy at any time t is denoted $E(t)$. The SM does not leak energy over time.

5. The ED-H power management strategy

In that section we describe the novel ED-H scheduler which was first presented and evaluated in [3]. Conventional EDF is a greedy scheduler since it executes jobs as soon as possible and spends the stored

energy disregarding needs of future occurring jobs. In that version of EDF called EDS (Earliest Deadline as Soon as possible), the processor is never let inactive if at least one job is awaiting for execution. If we assume jobs to be scheduled according to the earliest deadline rule, energy starvation on a job say τ_i can be only caused by a job, say τ_j which executes before the release of τ_i such that $d_j > d_i$. Energy starvation of τ_i caused by τ_j such that $d_j \leq d_i$ could not be avoided. It is obvious that clairvoyance on future jobs arrivals and future energy production will help the online EDF scheduler to anticipate possible energy starvation and deadline violation. Consequently, the main idea of ED-H is to authorize job executions only if no future starvation can occur.

5.1. Slack energy concept

We introduce the *static slack energy* of τ that represents the additional energy that could be consumed from any instant while still satisfying all the energy and timing constraints of τ. The static slack energy of a job set τ is given by $SSE_\tau = \min_{0 \leq t_1 < t_2 \leq d_{Max}} SSE_\tau(t_1, t_2)$.

We define the so-called *preemption slack energy* for current time t_c as the maximum energy that could be consumed by the currently active job that still guarantees the feasibility of higher priority jobs that may preempt it.

Let us define the slack energy of a job τ_i at current time t_c as $SE_{\tau_i}(t_c) = E(t_c) + E_p(t_c, d_i) - g(t_c, d_i)$ where $g(t_c, d_i)$ refers to the energy demand between t_c and d_i. Clearly, $SE_{\tau_i}(t_c)$ represents the maximum energy that could be consumed within $[t_c, d_i)$. If there is some job τ_i such that $SE_{\tau_i}(t_c) = 0$, then the execution of any job with deadline after d_i between t_c and d_i will lead to energy starvation for τ_i.

Let d be the deadline of the active job at current time t_c. We define the preemption slack energy of a job set τ at t_c as $PSE_\tau(t_c) = \min_{t_c < r_i < d_i < d} SE_{\tau_i}(t_c)$.

5.2. Slack time concept

Let us define *the processor demand of a job set* τ on the time interval $[t_1, t_2)$ as the total processing requirement between t_1 and t_2, denoted by $h(t_1, t_2)$. Then, we can define the *static slack time a job set* τ on the time interval $[t_1, t_2)$ as follows: $SST_\tau(t_1, t_2) = t_2 - t_1 - h(t_1, t_2)$. Clearly, $SST_\tau(t_1, t_2)$ represents the longest time that could be made available within $[t_1, t_2)$ after executing jobs of τ with release time at or after t_1 and deadline at or before t_2.

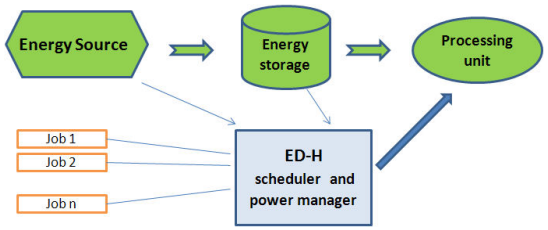

Figure 2. ED-H: a scheduler and a dynamic power manager

The slack time of a job set τ at current time t_c is $ST_\tau(t_c) = \min_{d_i > t_c} ST_{\tau_i}(t_c)$. The slack time represents the maximum continuous processor time that could be available from time t_c while still guaranteeing the deadlines of all the jobs.

Roughly speaking, ED-H will consist in permitting the processor to be inactive if the slack time is positive. In contrast the processor should imperatively start executing any job when the slack time falls to zero. In addition, a positive preemption slack energy signifies that the currently active job can continue execution. In contrast a null preemption slack energy imposes to stop execution and start recharging the energy storage unit. We are now prepared to describe the ED-H scheduling algorithm (see Fig.2).

5.3. Scheduling framework

Let $L_r(t_c)$ be the list of uncompleted jobs ready for execution at t_c. The ED-H scheduling algorithm obeys the following rules:

- **Rule 1:** The EDF priority order is used to select the future running job in $L_r(t_c)$.
- **Rule 2:** The processor is imperatively idle in $[t_c, t_c + 1)$ if $L_r(t_c) = \emptyset$.
- **Rule 3:** The processor is imperatively idle in $[t_c, t_c + 1)$ if $L_r(t_c) \neq \emptyset$ and either $E(t_c) = 0$ or $PSE_\tau(t_c) = 0$.
- **Rule 4:** The processor is imperatively busy in $[t_c, t_c + 1)$ if $L_r(t_c) \neq \emptyset$ and either $E(t_c) = C$ or $ST_\tau(t_c) = 0$
- **Rule 5:** The processor can equally be idle or busy if $L_r(t_c) \neq \emptyset$, $0 < E(t_c) < C$, $ST_\tau(t_c) > 0$ and $PSE_\tau(t_c) > 0$.

The algorithm says that:

- the processor must be inactive if either the energy storage unit is deplenished or executing any job

would prevent at least one future occurring job from being executed timely because of energy starvation i.e. the system has no preemption slack energy at t_c.

- the processor cannot be inactive if either the energy storage unit is fully replenished or making the processor idle would prevent at least one job from being executed timely because of time starvation i.e. the system has no slack time at t_c.

- the scheduler may decide on the processor state when the storage unit is neither full nor empty and the system has both slack time and preemption slack energy.

- we start charging the storage unit when, either it is empty or there is not enough energy to guarantee the feasible execution of all future occurring jobs.

6. Properties of ED-H

6.1. ED-H optimality

Theorem 1: The ED-H scheduling algorithm is optimal for the RTEH model.
Proof: see [3]

Optimality signifies that ED-H can produce a valid schedule

- if there is no time interval with a length lower than the processor demand
- and there is no time interval where the energy demand is greater than the available energy.

6.2. ED-H schedulability test

Hereafter, we present a test for verifying that a given job set can indeed meet its deadlines, be given the capacity of the energy storage unit and the profile of power drawn from the environment. We give a necessary and sufficient condition for ED-H schedulability. As ED-H is optimal, the condition is also a feasibility condition.

Theorem 2: τ is feasible if and only if

$$SST_\tau \geq 0 \text{ and } SSE_\tau \geq 0 \quad (1)$$

Proof: see [3]

The objective of feasibility checking is to predict whether time and energy will be enough to meet the timing requirements of all the jobs. In the design of real-time systems composed of well known periodic tasks with no energy limitations, we perform an off-line check and we use an online algorithm to schedule and dispatch the jobs at runtime. For RTEH systems, the checking can be done off-line only when first the jobs are instances of periodic tasks and second the energy profile is precisely characterized for all the application lifetime. In all other situations, the schedulability checking should be realized at runtime in dependence with the horizon of the prediction technique. This signifies that regularly, the schedulability test is performed in order to verify that all the jobs released on the next time window will be feasibly scheduled. Otherwise, a decision must be made in order to make the system feasible by discarding some jobs and consequently get a lesser Quality of Service.

7. Implementation considerations

We wait from the real-time energy harvesting system to achieve energy-neutral operation i.e. to use the harvested energy at an appropriate rate such that the system continues to operate perennially. As the harvested energy availability varies with time in a non deterministic manner, monitoring the residual capacity is not sufficient. We need a sophisticated characterization of the energy source. The ED-H scheduler not only must track the generated energy, but also the energy flow into and out of the storage unit to provide an accurate estimate of the residual capacity. If the ED-H scheduler as well as the schedulability test are used for practical implementation, the first required measurement is the amount of extracted environmental energy, which is a technological difficulty. Nonetheless, various prediction models have been studied and can be found in [10].
A slack computation algorithm is correct if it never says that the system has slack when it has not, as regards time or energy. Indeed, this may cause a job to complete too late due to time or energy insufficiency. Computations of slack time and slack energy by the scheduler at runtime can use a static or dynamic approach depending on the processing load. If all the jobs are issued from a periodic task set, the initial slacks of all the jobs (known beforehand) can be computed off-line based on the given parameters relating to time and energy. It can be verified that the slacks are calculated in $O(n^2)$ where n is the total number of jobs. The scheduler just updates the slacks during runtime (see [2]) for slack time computation). According to the dynamic approach, the scheduler computes the slack time and the slack energy at run time from scratch.

Dynamic computation of the slack time and the slack energy present the advantage to reclaim processor time or/and energy not used by the jobs. We keep track of the cumulative unused processor time and unconsumed energy. However, this leads to high runtime overheads.

8. Concluding remarks

In this work, we proposed a new general model dedicated to energy harvesting real-time systems. We studied first the problem of checking the feasibility for such systems and second an optimal scheduling strategy for the real-time jobs with processing and energy requirements.

The ED-H scheduler is a semi-online variant of the famous EDF scheduler. Similarly to LSA, the ED-H power management strategy is optimal but it is more flexible than LSA since the user may decide when inserting idle periods for recharging the storage unit (as long as slack time is available and no energy is wasted because of energy overflow).

Such scheduler is highly flexible since it permits us to schedule time critical jobs which are periodic, sporadic or aperiodic ones.

Future work includes the adaptation of ED-H to new generation processors with DVFS (Dynamic Voltage Scaling) facilities.

References

[1] Y. Abdeddaim, Y. Chandarli, D.Masson. *The Optimality of PFPasap Algorithm for Fixed-Priority Energy-Harvesting Real-Time Systems*, Proc. of 25th Euromicro Conference on Real-Time Systems, 2013.

[2] H. Chetto, M. Chetto. Some Results of the Earliest Deadline Scheduling Algorithm. *IEEE Transactions on Software Engineering*, Volume 15, Issue 10, pp. 1261-1270, 1989.

[3] M. Chetto. Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems. *IEEE Transactions on Emerging Topics in Computing*, DOI: 10.1109/TETC.2013.2296537, 2014.

[4] H. El Ghor, M. Chetto, R. Hage Chehade. *A real-time scheduling framework for embedded systems with environmental energy harvesting*, Journal of Computers and Electrical Engineering, Volume 37 Issue 4, pp. 498-510, 2011.

[5] H. El Ghor, M. Chetto and R. Hage Chehade. "A Nonclairvoyant Real-Time Scheduler for Ambient Harvesting Sensors". International Journal of Distributed Sensor Networks, Volume 2013, Article ID 732652, 11 pages, http://dx.doi.org/10.1155/2013/732652.

[6] A. Kansal, J. Hsu. *Harvesting aware Power Management for Sensor Networks*, Proc. of ACM/IEEE Design Automation Conference, pp. 651-656, 2006.

[7] A. Kansal, J. Hsu, S. Zahedi, M.B. Srivastava. *Power Management in Energy Harvesting Sensor Networks*, ACM Transactions on Embedded Computing Systems, Vol. 6, No. 4, 2007.

[8] J. W. S. Liu. *Real-Time Systems*, Prentice Hall, 592 pages, 2000.

[9] C.-L. Liu, J.-W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. of the Association for Computing Machinery*, 20(1), pp. 46-61, 1973.

[10] S. Liu, J. Lu, Q. Wu, Q. Qiu. *Harvesting-Aware Power Management for Real-Time Systems with Renewable Energy*, IEEE Transactions on Very Large Scale Integration Systems, pp. 1-14, 2011.

[11] C. Moser, D. Brunelli, L. Thiele, L. Benini. *Real-time scheduling for energy harvesting sensor nodes*, Real-Time Systems, Volume 37, Issue 3, pp. 233-260, 2007.

[12] S. Priya, D.-J. Inman. *Energy Harvesting Technologies*, Springer-Verlag, New York (USA), 2009.